A Verification Method for a Commitment Strategy of the BDI Architecture

Tadashi ARARAGI\textsuperscript{a,1} Shiro TAKATA\textsuperscript{b,2} NIDE Naoyuki\textsuperscript{c,3}

\textsuperscript{a} NTT Communication Science Laboratories
Hikaridai 2–4, Seika-cho, Soraku-gun, Kyoto, 619-0237 Japan
\textsuperscript{b} ATR Media Information Science Laboratories
Hikaridai 2–2–2, Seika-cho, Soraku-gun, Kyoto, 619-0288 Japan
\textsuperscript{c} Nara Women’s University
Kita-Uoya Nishimachi, Nara-shi, Nara, 630-8506 Japan

Abstract
We present a method to solve a verification problem that arises in implementing a commitment strategy for the BDI architecture. This problem introduces a new aspect of verification such that a state transition depends on a verification done at each state. We formalize this problem and give a decision procedure for the verification.

1 Introduction
In this paper, we present a method to solve a new verification problem that arises when implementing a commitment strategy[6] for the BDI architecture [8,9].

The BDI architecture is a model of agent systems, and introduces the notions of belief, desire, and intention to realize flexible plan descriptions for agents. One important notion in the BDI architecture is its commitment strategy. A commitment strategy requires agents to hold their intentions as long as the intended affairs are believed possible in the future. In other words, agents give up their current intentions if they find the intentions cannot be achieved in the future. This strategy makes the agent’s plans more concise and execution of the plans more efficient. In order to implement a commitment strategy, especially the type called single-minded [6], we need a method to predict the feasibility of one or more intentions.

\textsuperscript{1} Email: araragi@cslab.kecl.ntt.co.jp
\textsuperscript{2} Email: shiro@atr.co.jp
\textsuperscript{3} Email: nide@ics.nara-wu.ac.jp

©2002 Published by Elsevier Science B. V. Open access under CC BY-NC-ND license.
There have been many implementations that reflect the deduction processes of belief, desire and intention. Using the deduction mechanisms of BDI logic [7,5], we can satisfactorily implement these processes so as to execute the plans. Unfortunately, the single-minded commitment strategy has not been fully implemented. There are other works that deal with commitment within the frameworks of temporal logics and dynamic logic. But these works address, at present, the analysis of the behavior of BDI agents, and provide no practical guide to the implementation of a commitment strategy for the BDI architecture.

The aim of this paper is full implementation of a commitment strategy of the single-minded type. We deal with the problem of predicting the feasibility of intentions as a verification problem of agent programs. This verification problem has a complicated recursive structure, because an agent’s behavior in the future also depends on an assessment of its intentions at that point. Thus, this verification problem is quite different from the conventional verification problems [4,1,2]. In this paper, we call this type of verification commitment verification.

In this paper, section 2 introduces this problem informally and then formalizes it by introducing the notion of connection/disconnection valuation to a state transition tree of agent programs. Following this section, some typical examples of the valuations are given in Section 3, and we discuss the valuations. Section 4 describes a decision procedure for this problem based on the tableaux method, and the correctness of the procedure is shown in section 5.

2 Commitment Verification

2.1 Informal Introduction of Commitment Verification

Many implementations of the BDI architecture have been proposed. We can summarize the essence of these implementations in the following form by ignoring the characteristics of each implementation.

- A state of a BDI agent is a set of BDI formulas
- A program of a BDI agent is a set of plan rules where each rule has a precondition to check the current state of the agent, as well as actions which are executed if the result of the check is successful.

We give an example of plan rules below.

\[
\text{BEL}(\text{realize}(agent_1, \phi_1)) \& \text{INTEND}(\phi_1) \iff \text{send}(agent_1 : \text{request}(\phi_1)), \text{add}(\text{INTEND}(\phi_2))
\]

The meaning of this rule is the following. If the agent holding this rule, we call it \textit{agent}_0, believes that \textit{agent}_1 can realize \phi_1, and intends to perform \phi_1, then \textit{agent}_0 sends a message to \textit{agent}_1, that asks \textit{agent}_1 to achieve \phi_1 on behalf of
agent₀, and agent₀ creates a new intention, \( \phi_2 \), to support agent₁ in achieving \( \phi_1 \).

If the current state of agent₀ is, for example,

\[
\{ \text{BEL}(\text{reliable}(\text{agent}_1)), \text{BEL}(\text{reliable}(x) \rightarrow \text{realize}(x, \phi_1)), \text{INTEND}(\phi_1) \},
\]

then the rule is enabled. Each agent selects one enabled rule and executes it. If agent₀ selects this enabled rule, the state becomes

\[
\{ \text{BEL}(\text{reliable}(\text{agent}_1)), \text{BEL}(\text{reliable}(x) \rightarrow \text{realize}(x, \phi_1)), \\
\text{INTEND}(\phi_1), \text{INTEND}(\phi_2) \}.
\]

By employing a theorem prover of BDI logic [5], we can realize the mental processes of belief, desire and intention in this framework. What then about commitment? In this example, if we consider a commitment strategy, especially single-minded commitment, we have to judge if \( \phi_1 \) will be possible in the future. There can be various criteria to judge this possibility. In this paper, we take a formal criterion. That is, we calculate the state transition graph of an agent from the given program and the current state, and check if \( \phi_1 \) is possible in the future. This is reduced to a problem of program verification. However, there is a big and important difference between this kind of verification and ordinary program verification. In the state transition of BDI agents, the commitment strategy is applied to each transition and, in this application, we need another preemptive verification. That is, each transition depends on the result of another verification. It is obvious that the application of verification for commitment has a recursive structure. The first problem to realizing this verification is how to formalize it.

2.2 Formalization of Commitment Verification

2.2.1 Agent States and Agent programs

• An agent state is a finite set of BDI logic formulas.

• An agent program is a finite set of plan rules.

A plan rule has the following form: 

\[
\text{precondition}_1, \text{precondition}_2 \implies \text{action}_1, \text{action}_2, \cdots
\]

\( \text{precondition}_1 \) and \( \text{precondition}_2 \) are propositional BDI logical formulas. The validity of these formulas is checked wrt the current state of an agent. If \( \text{precondition}_1 \) is valid, and \( \text{precondition}_2 \) is not, then we say the rule is enabled at the state in question. \( \text{precondition}_2 \) plays a role of representing negation in a precondition of a plan rule. The following is an example of plan rules using a \( \text{precondition}_2 \).

\[
\text{BEL}(\phi_1), \text{INTEND}(\phi_2) \implies \text{add}(\text{INTEND}(\phi_3))
\]
This rule means that $\phi_2$ and $\phi_3$ are exclusive as intentions.

There are two types of actions. The first action modifies its agent’s state such as $\text{add}(\phi)$ and $\text{rm}(\phi)$ which mean adding logical formula $\phi$ to the current state and removing $\phi$ from it, respectively. The second interacts with external worlds such as sending messages to other agents and changing the state of worlds commonly observable by agents.

[Conventions for simplicity]
For simplicity, we impose two restrictions on the plan rules.

(1) Each precondition is a \textit{positive BD formula} or a conjunction of a positive BD formula and an \textit{I-principal formula}. Here, a positive BD formula is that constructed from $\text{BEL}$, $\text{DESIRE}$, $\&$, $\text{or}$ and propositional variables, and an I-principal formula is that with the form $\text{INTEND}(\text{positive BD formula})$. To make the discussion on commitment verification simpler, we also assume that an I formula appears in at most one of $\text{precondition}_1$ and $\text{precondition}_2$ in a given rule. If $\text{INTEND}(\phi)$ appears in $\text{precondition}_1$ or $\text{precondition}_2$ as a part of the conjunction, we call $\phi$ or $\lnot\phi$ a support formula of the rule respectively.

(2) The plan rules do not include actions that involve interaction with external worlds. Dealing with such interactions would bring nondeterminism to the state transition of agents. We can deal with this nondeterminism in verification but it makes the argument complicated. So, we will omit this part in this paper.

2.2.2 Preemptive State Transition Graph and Tree

Definition: We first define the preemptive state transition graph ($\text{pe graph}$ for short) for a rule $r$ and a state $s$, in other words, we will show how to construct the pe graph. We first make the initial node $n$ corresponding to $s$ and the initial edge $e$ corresponding to $r$. For the end node $n'$ of $e$, we attach the state resulting from applying actions of $r$ to $s$. Next, let rule $r_1, \ldots, r_p$ be the enabled rules at the state attached to $n'$. We create $p$ pieces of edges from $n'$ corresponding to each $r_i$, and attach the state resulting from applying the rule of the edge to $n'$, to the end node. We repeatedly apply this procedure to newly created nodes. Here, we identify two nodes with each other if their attached states are the same; this yields a finite graph. A \textit{preemptive state transition tree} ($\text{pe tree}$ for short) for rule $r$ and state $s$ is that obtained by expanding the pe graph for $r$ and $s$ into a tree form. See Fig. 1.

A map which assigns value “$C$” or “$D$” to each edge of a pe tree $T$ is called a \textit{valuation} of $T$. $C$ means “connected” and $D$ does “disconnected” for the edge, respectively. $C$ also means that the rule corresponding to the edge is executed at that state.

We prepare some notations for pe trees. Let $e$ be an edge of a pe tree, and $n$, $n_1$ and $n_2$ be nodes of a pe tree. $\text{end}(e)$ is the end node of $e$. $\text{spf}(e)$ is the
support formula of the plan rule corresponding to $e$. $g\_id(n)$ and $g\_id(e)$ are the graph node id of $n$ and the graph edge id of $e$ in the original pe graph respectively. $[n_1, \ldots, n_2]$ is the path from $n_1$ to $n_2$ in a pe tree. $(n_1, \ldots, n_2)$ is that not including $n_1$, and $[n_1, \ldots, n_2]$ is that not including $n_2$.

We explain the role of values $C$ and $D$ intuitively and why this commitment verification is very different from conventional verifications. Let an agent be at a state $st$ and thinks to execute a plan rule which includes an intention $I(\phi_1)$ in its predondition$_1$. Let the corresponding transition be $e_1$, that is, $spf(e_1)=[\phi_1]$. If the agent obeys the commitment strategy, it has to judge if $\phi_1$ is possible or not in the future from $st$. Then the agent expands its pe tree from $st$ based on its agent program. Let’s assume that the agent found there is the only one state $st1$ where $\phi_1$ is possible in the pe tree. Can the agent think $\phi_1$ is possible in the future? At this moment, we cannot say “yes”. Let’s assume that there is a transition $e_2$ from a state $st2$ between $st$ and $st1$, whose corresponding plan rule depends on another intention $I(\phi_2)$. If the agent judges that $\phi_2$ is possible in the future from $st2$, then the transition $e_2$ is connected (labeled $C$), and $\phi_1$ is possible in the future from $st$, provided that there is no other transition depending on a intention between $st$ and $st1$. Otherwise, $e_2$ is disconnected (labeled $D$) and $\phi_1$ is impossible in the future from $st$.

What makes the verification more complicated is that the judgment at $st2$ also depends on connection and disconnection of transitions including intentions, and again the connection and disconnection are involved in judgments for the futures from those states.
To formalize the relation between the connection/disconnection and the judgment, we introduce the notion of consistent valuation below.

**Definition:** Let $T$ be a pe tree and $V$ be a valuation of $T$. The subtree of an edge $e$ wrt $V$ is a subtree of $T$ whose root is the end node of $e$, and whose nodes are those of $T$ that can be reached from the root through edges whose value is $C$ by $V$. We denote the subtree by $T(e, V)$.

**Definition:** Let $T$ be a pe tree, $e$ be an edge of $T$, and $V$ be a valuation of $T$. Let $[\phi]$ or $\neg [\phi]$ be the support formula of $e$. We write $T(e, V) \models \text{EF} \phi$ if for some node $n$ in $T(e, V)$, $n \vdash \phi$ holds. Here $n \vdash \phi$ means the agent state attached to $n$ derives $\phi$ logically. We also write $T(e, V) \models \neg \text{EF} \phi$ if otherwise.

**Definition:** Let $T, V$ and $e$ be as in given in the above definition. We say $e$ is satisfied by $V$, if one of the following conditions is satisfied.
- the case that $\text{spf}(e) = [\phi]$
  $V(e) = C$ and $T(e, V) \models \text{EF} \phi$, or $V(e) = D$ and $T(e, V) \models \neg \text{EF} \phi$
- the case that $\text{spf}(e) = \neg [\phi]$
  $V(e) = D$ and $T(e, V) \models \text{EF} \phi$, or $V(e) = C$ and $T(e, V) \models \neg \text{EF} \phi$

Note that the combination of $\text{spf}(e) = \neg [\phi]$ and $V(e) = D$ gives an eventual property. We say $V$ is a consistent valuation of $T$, if every edge of $T$ is satisfied by $V$.

Now we come to the formal definition of verification for commitment.

**Definition:** Plan rule $r$ is commitment-enabled at state $s$, if the pe tree for $r$ at $s$ has a consistent valuation whose value for the initial edge of $T$ is $C$.

## 3 Examples and Discussion

In this section, we give some typical examples of pe graphs, pe trees, and their valuations. Example (a) in Fig. 2 shows a pe graph and the associated pe trees with different valuations. The pe tree is finite in this case. The pe graph shows that $\phi_1$ is valid at $\text{end}(e_2)$, and $\phi_2$ and $\phi_3$ are valid at $\text{end}(e_3)$, as well as $\text{spf}(e_1) = [\phi_1]$, $\text{spf}(e_2) = \neg [\phi_2]$ and $\text{spf}(e_3) = [\phi_3]$. In the left pe tree, we first set $V(e_1) = C$ for the initial edge $e_1$. As a result, to make the valuation consistent, $V(e_2)$ must be $C$, because $V(e_1) = C$ and $\text{spf}(e_1) = [\phi_1]$ require $T(e_1, V) \models \text{EF} \phi_1$, and $\text{end}(e_2) \vdash \phi_1$. Next, to set $V(e_2) = C$ in a consistent valuation, $V(e_3)$ must be $D$, because $V(e_2) = C$ and $\text{spf}(e_2) = \neg [\phi_2]$ require $T(e_2, V) \models \neg \text{EF} \phi_2$, and $\text{end}(e_3) \vdash \phi_2$. On the other hand, $\text{end}(e_3) \vdash \phi_3$ implies $T(e_3, V) \models \text{EF} \phi_3$ and $\text{spf}(e_3) = [\phi_3]$. These facts yield $V(e_3) = C$ and contradict with the result above. Therefore, there is no consistent valuation with $V(e_1) = C$. The valuation depicted in the right pe tree is easily checked to
be consistent. Thus, the plan rule corresponding to edge $e_1$ is not commitment-enabled, and the agent does not execute it at this moment.

Example (b) shows a pe graph whose associated pe tree is infinite. This pe tree shows $g_{id}(e_1') = e_1$, $g_{id}(e_2') = e_2$, . . . , and $end(e_1) \vdash \phi_1$ and $end(e_2) \vdash \phi_2$, as well as $spf(e_1) = \neg [\phi_1]$ and $spf(e_2) = \neg [\phi_2]$. In this example, we can obtain a consistent valuation with $V(e_1') = C$ by setting $V(e_i') = C$ if $g_{id}(e_i') = e_1$ and $V(e_i') = D$ if $g_{id}(e_i') = e_2$ uniformly, while we cannot in example (c). In example (c), if we set $V(e_1') = C$, then $V(e_2') = C$ is required by the same discussion above. In the same way, $V(e_2') = C$ requires $V(e_3') = D$, $V(e_3') = D$ does $V(e_4') = D$, and $V(e_4') = D$ does $V(e_5') = D$. After this, we obtain a consistent valuation by executing the remaining settings periodically. This example shows why we have to introduce pe trees, in addition to pe graphs, to discuss the preemptive decision properly. Even if two agent states are the same, we can take different actions on them if they are seen to occur at different times in the future. In example (d), we can give a consistent valuation that gives value $D$ to the initial edge, but not value $C$.

In example (c), we mentioned periodicity of a valuation. In general, there is some periodicity in every consistent valuation, but it is a little complicated and we need more notions to formalize it, which we will introduce in section 4. In section 5, we will also show how the periodicity is located in pe trees.

As we pointed out in the introduction, the commitment verification is very different from conventional verifications.
4 Decision Procedure for Commitment Verification

We prepare some definitions to introduce a decision procedure for commitment verification.

Definition: Let $T$ be a pe tree and $n$ be a node of $T$. The period of $n$ is the minimum distance from $n$ to the successor nodes of $n$ that have the same pe graph node id with $n$. The period of $T$ is the maximum value of period of $n$ when $n$ is varied among all nodes in $T$. It is clear that the period of $T$ is finite.

Definition: A request set for an edge of a pe tree $T$ is a pair of sets: $\mathcal{EF}$-set and $\mathcal{¬EF}$-set. $\mathcal{¬EF}$-set is a set of formulas whose element is $\phi$ for some support formula $\phi$ or $\mathcal{¬}[\phi]$. $\mathcal{EF}$-set is the list of pe graph node ids whose length is at most the period of $T$. It is clear that the number of possible request sets is finite when $T$ is given.

Definition: A preemptive index ($\text{peindex}$ for short) for edge $e$ is a triple of a graph node id of $\text{end}(e)$, a request set attached to $e$ and a value $C$ or $D$ given to $e$. We denote an upper bound of the number of possible pe indexes by $M$.

In the following, we introduce a decision procedure for commitment verification. We hereafter call this procedure $cm$-verification. When an agent program, the current state of the agent, and an addressed plan rule are given for $cm$-verification, we can create a pe graph in a canonical way. We can then expand the pe tree $T$ from the pe graph. In the decision procedure, we give a value $C$ or $D$ to each edge of $T$ incrementally from the root to its successors. We efficiently locate the consistent valuation in parallel using the tableaux method. A branch in the tableau is a partial valuation of $T$, and carries request sets for each edge to guarantee the consistency of the valuation. There are two points to these calculations. The first is that, to guarantee the eventual property (expressed in $\mathcal{EF}$-set), we look for the witness only at some fixed distance: the period of $T$ denoted by $L$. The second is that, to guarantee the inevitable property (expressed in $\mathcal{¬EF}$-set), we check the fixed depth of the tree: the depth equals a kind of period of valuation given as value $L \ast M$. We give reasons for these points in section 5.

step 1: The tableau $tab := [br_0]$, where $br_0$ is an initial edge of $T$, its value is $C$, and its $\mathcal{EF}$-set and $\mathcal{¬EF}$-set are empty.

step 2: If $tab$ is empty, then return “no” and stop this procedure.

Take a branch $br$ out of $tab$ fairly.
Select a leaf edge $e$ from $br$ fairly.
step 3: Let tmp_tab1 be an empty list.

If $e$ has value $C$ and $spf(e) = [\phi]$, or $e$ has value $D$ and $spf(e) = \neg[\phi]$, then find nodes $n$ satisfying the following conditions.

- $n$ is a successor of $e$, and the distance from $\text{end}(e)$ to $n$ is not more than $L$.
- $n \vdash \phi$, and for every node $n'$ in $[\text{end}(e), \cdots, n]$, $n'$ satisfies $n' \not\vdash \phi$
- for every $n'$ in $(\text{end}(e), \cdots, n]$ and every $\phi'$ in $\neg\text{EF-set}(e)$, $n'$ satisfies $n' \not\vdash \phi'$

Let $n_1, \ldots, n_p$ be the nodes that satisfy this condition. If there is no such a node, remove $br$ from $tab$ and go to Step 2.

Make $p$ copies of $br : br_1, \ldots, br_p$. For each $br_i$, add the list $[g_{.id}(\text{end}(e)), \cdots, g_{.id}(n_i)]$ to $\text{EF-set}(e)$.
Add all $br_i$ to $tmp\_tab1$.

If $e$ has value $C$ and $spf(e) = \neg[\phi]$, or $e$ has value $D$ and $spf(e) = [\phi]$, then add $\phi$ to $\neg\text{EF-set}(e)$.
Add $br$ to $tmp\_tab1$.

step 4: Apply the procedures (1) and (2) to every element in $tmp\_tab1$.

(1) Check if for every $\phi'$ in $\neg\text{EF-set}(e)$, $\text{end}(e)$ satisfies $\text{end}(e) \not\vdash \phi'$. If this check fails, remove $br$ from $tmp\_tab1$.

(2) Remove the head element of each list in $\text{EF-set}(e)$. If a list becomes empty as a result, we remove the list from $\text{EF-set}(e)$.

If $tmp\_tab1$ becomes empty, go to Step 2.

step 5: 5.1 Let tmp_tab2 be an empty list.
Apply the following procedure to every element in $tmp\_tab1$.

For each edge $e'$ starting at $\text{end}(e)$ in $T$, if $\text{end}(e')$ has the same graph id with some head element of a list in $\text{EF-set}(e)$, then give value $C$ to $e'$.

Let $e_1, \ldots, e_q$ be the edges starting at $\text{end}(e)$ in $T$, and that do not satisfy the above condition. Next, make $2^q$ copies of $br$ and give value $C$ or $D$ to each $e_i$ so that all possibilities are exhausted.
Then add all the newly created branches to $tmp\_tab2$.

5.2 Apply the following procedure to every element in $tmp\_tab2$.
For each edge $e'$ starting at $\text{end}(e)$ in $T$,

if its value is $C$, we set $\neg\text{EF-set}(e')$ to $\neg\text{EF-set}(e)$, and $\text{EF-set}(e')$ to the set of lists in $\text{EF-set}(e)$ whose head element is the same as the $g_{.id}$ of $\text{end}(e')$.
else if its value is $D$, set $\text{EF-set}(e')$ and $\neg\text{EF-set}(e')$ to empty.
Fig. 3. A neighbor path of \((n'_1, n_1)\) for \(e\)

step 6: If \(br\) at Step 2 has no change in the procedures from Step 2 to Step 5, that is, \(br\) is fully expanded, then return “ok” and stop this procedure. If for some branch in \(tmp\_tab2\), the shortest length from the root to the leaf nodes of \(br\) is more than \(L \ast M\), then return “ok” and stop this procedure. Otherwise, append \(tmp\_tab2\) to \(tab\) and go to Step 2.

In this procedure, we assume that every edge has a support formula. Actually, some rule may not depend on any intention. In this case, the corresponding edge does not have a support formula, and we give value \(C\) to the edge.

Because the size of branches appearing in this procedure is bounded by the termination condition at Step 6, this procedure finally terminates.

5 The Correctness of cm-Verification

Definition: Let \(T\) be a pe tree, \(e\) be an edge of \(T\), \(n_1\) be a successor of \(e\), and \(L\) be the period of \(T\). If the distance of \(n_1\) from the end node of \(e\) is more than \(L\), there are nodes \(n_2, n'_1, n'_2\) that satisfy the following conditions.

1. \(n_1\) and \(n_2\) have the same graph id, and the distance of \(n_2\) from the end node of \(e\) is equal to or less than \(L\).
2. \(n'_1\) and \(n'_2\) have the same graph id, \(n'_2\) is on the path from the end node of \(e\) to \(n_1\), and for the path from \(n'_1\) to \(n_1\): \([n'_1, \ldots, n_1]\) and the path from \(n'_2\) to \(n_2\): \([n'_2, \ldots, n_2]\), \(g\_id(n'_1), \ldots, g\_id(n_1)\) is equal to \(g\_id(n'_2), \ldots, g\_id(n_2)\), where \(g\_id(n)\) is the graph id of \(n\).

We call \((n'_2, n_2)\) a neighbor path of \((n'_1, n_1)\) for \(e\). See Fig. 3.

Definition: Let \(T\) be a pe tree and \(V\) be a consistent valuation of \(T\). For any edge \(e\) of \(T\), if \(e\) has support formula \([\phi]\) or \(\neg[\phi]\), and \(e\) is given value \(C\) or \(D\) by \(V\) respectively, then, by the consistency of \(V\), there is a node \(n\) such that \(n\) can be reached from the end node of \(e\) through only edges whose value is \(C\) in \(V\), and \(n\) satisfies \(n \vdash \phi\). Let \(n_1\) be a nearest node from \(e\) that satisfies the
above condition. If the distance from the end node of \(e\) to \(n_1\) is more then the period of \(T\), we take nodes \(n_2, n'_2\) and \(n'_1\) such that \((n'_2, n_2)\) is a neighbor path of \((n'_1, n_1)\) for \(e\). Let \(e_2\) be the first edge having value \(D\) in the path from \(n'_2\) to \(n_2\). Let \(e_1\) be the corresponding edge, at the same position in other words, in the path from \(n'_1\) to \(n_1\). We then replace the valuation under edge \(e_2\) with that under edge \(e_1\) (see Fig. 4). We call this replacement the modification of \(V\) by \(e\).

**Lemma:** Let \(T\) be a pe tree, \(e\) be an edge of \(T\), and \(V\) be a consistent valuation of \(T\). The valuation obtained by the modification of \(V\) by \(e\) is also a consistent valuation of \(T\).

**Proof:** We use the notations in the above definition. Let’s assume that the modification gives an inconsistent valuation. Then there is edge \(e'\) that is an ancestor of \(e_2\) and that is not satisfied by the modified valuation \(V'\). We only consider the case that \(\text{spf}(e') = [\phi]\). The other case is similar. If the value of \(e'\) by \(V\) is \(C\), this case does not give rise to any inconsistency, because the modification expands the set of nodes connected to \(e'\). If the value is \(D\), then there is node \(n\) in the subtree under \(e_2\) that satisfies \(n \vdash \phi\), and \(n\) is connected to \(e\). Moreover, \(e'\) is an ancestor of \(e\) because \(e\) and \(e_2\) are connected, that is, all edges between them have value \(C\). Let \(n'\) be the node corresponding to \(n\) in the subtree under \(e_1\). Because \(e'\) is connected to \(n\), \(e_1\) is connected to \(n'\), and thus connected to \(e\). This means \(V\) was already inconsistent.

**Definition:** Valuation \(V\) of pe tree \(T\) is \(L\)-consistent, if \(V\) is a consistent valuation of \(T\), and for every edge \(e\) in \(T\), if \(e\) has value \(C\) or \(D\) by \(V\) and \(\text{spf}(e) = [\phi]\) or \(\neg [\phi]\) respectively, then there is a successor node \(n\) of \(e\) such that \(n\) is connected to \(e\) in \(V\), the distance between the end node of \(e\) and \(n\) is not more than \(L\), and \(n\) satisfies \(n \vdash \phi\).

**Claim:** If pe tree \(T\) has a consistent valuation, it has an \(L\)-consistent valuation.
Proof: When a consistent valuation is given, we repeatedly modify its edge from the root to its successors. The desired $L$-consistent valuation is the limit of this operation.

Theorem: A plan rule is commitment enabled, if and only if the decision procedure of cm-verification returns “ok”.

Proof:
($\Leftarrow$) If the procedure returns “no”, the pe tree doesn’t have a consistent valuation:
It is not hard to see that if the procedure returns “no”, there is no $L$-consistent valuation for the addressed pe tree $T$. Then, by the claim above, $T$ doesn’t have a consistent valuation.
($\Rightarrow$) If the procedure returns “ok”, the pe tree has a consistent valuation:
In the following, we construct a consistent valuation.
When the procedure returns “ok”, its tableau is not empty. We take a branch $br$ from the tableau.
Let $S_1$ be the set of nodes of $br$, whose distance from root is not more than $L$. Let $dS_1$ be the set of leaf nodes of $S_1$ in the usual sense. Let $S_2$ be the set of nodes of $br$, whose distance from a node in $dS_1$ is not more than $L$. Let $dS_2$ be the set of leaf nodes of $S_2$. In the same way, we define $S_3, S_4, \cdots$ and $dS_3, dS_4, \cdots$.

Lemma: For some $k$, the following is valid.
$\{m \mid m$ is a pe index of a node in $dS_k\} \subset \{m \mid m$ is a pe index of a node in $S_1 \cup \cdots \cup S_{k-1}\}$ —— (*)
Here, we attribute each pe index of an edge to its end node.

Proof: We check if (*) is valid, in the order $k = 2, 3, 4, \cdots$. If (*) is not valid for $k$, it means that $S_k$ includes a node whose pe index is different from those in $S_1 \cup \cdots \cup S_{k-1}$. In other words, a new pe index must emerge in $S_k$. But the number of possible pe indexes is finite. Then (*) is valid for $k$ less than $M$, the number used in the decision procedure. See Fig. 5 for $k = 4$. 

Fig. 5.Appearances of pe indexes in a branch

[Diagram of a branch with nodes and pe indexes labeled with $m_1, m_2, m_3, \ldots$ and $S_1, S_2, S_3, S_4$.]
We use Fig. 6 to explain how to construct a consistent valuation. The left tree in Fig. 6 shows a part of branch $br$ that satisfies the condition (*) in Lemma. In this case, we assume the pe indexes appearing in $dS_k$ are $m_1$ and $m_2$. Because of (*), in $S_1 \cup \cdots \cup S_{k-1}$ there are nodes with pe index $m_1$ and $m_2$. Let the nodes be $n_1$ and $n_2$ respectively. $n_1$ and $n_2$ yield two subtrees in $S_1 \cup \cdots \cup S_k$, whose roots are $n_1$ and $n_2$, and let the subtrees be $sT_1$ and $sT_2$, respectively. We make copies of $sT_1$ and $sT_2$, and connect them to the tree of $S_1 \cup \cdots \cup S_k$ by gluing the nodes in $dS_k$ to the corresponding roots of subtrees. We repeatedly glue these subtrees to the leaves of the operated tree: see the right hand side of Fig. 6.

It is not hard to see that this repetition gives a consistent valuation.

6 Conclusion and Future Work

In this paper, we have formalized a verification problem that arises in implementing a single-minded commitment strategy for the BDI architecture. We also gave a decision procedure for the verification. While there are other approaches to implementing a single-minded commitment strategy, we take a rather formal approach. This approach raises an interesting problem for program verification, such as a state transition for which verification is being considered depends on the verifications of other states. We are just at the start point of this unique problem and have to more fully elaborate the formalization and decision procedure. We also want to give a clear connection of our approach with the formalization of commitment strategies in temporal logics given in [6,3] so that we can discuss the correctness of our implementation of the commitment strategy in a logical framework.

Acknowledgments
We would like to thank Yoshifumi Ooyama, Norihiro Hagita and Kenji Mase for supporting this work. This research was also supported in part by the Telecommunications Advancement Organization.

References


